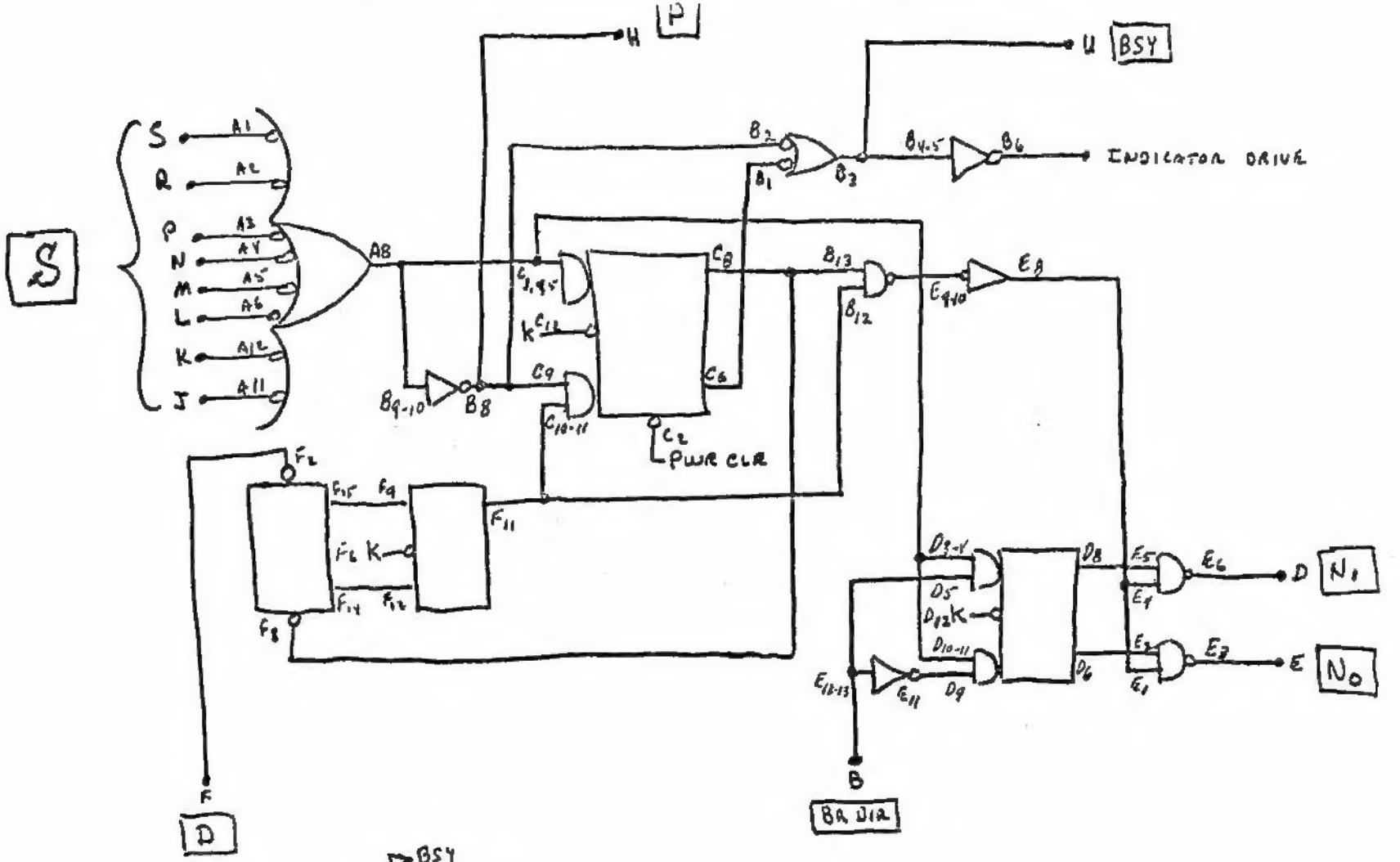
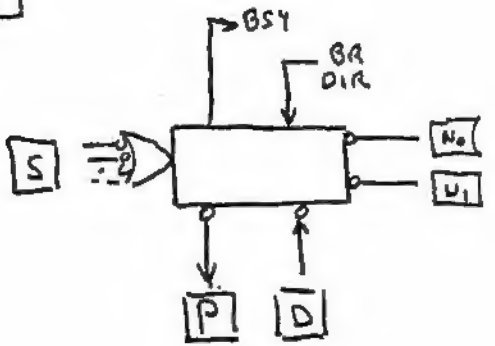
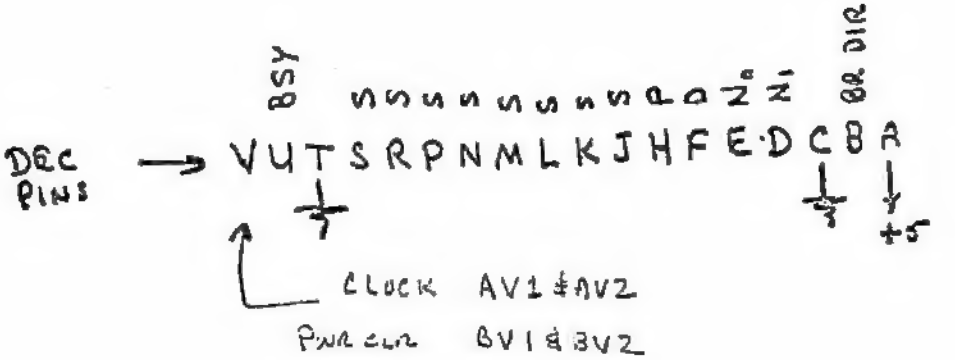




IC's

A	7430
B	7400
C	7472
D	7472
E	7404
F	7476



Tables from the original schematic:

ICs

A	7430
B	7400
C	7472
D	7472
E	7400
F	7476

DEC

pin	
V	AV1 & AV2: CLOCK; BV1 & BV2: PWR CLR
U	BSY
T	GND
S	S
R	S
P	S
N	S
M	S
L	S
K	S
J	S
H	P
F	D
E	N0
D	N1
C	GND
B	BR DIR
A	+5

[Editor's note: Each printed circuit board was the size of a DEC double-height, double-width "Flip Chip" module. These modules have two "paddles," and each paddle has 18 pins on each side. The four UCMs on a board used paddle A side 1, paddle A side 2, paddle B side 1, and paddle B side 2, respectively. As shown above, only the use of pin V differed between the 4 sets of pins.]

Digital System Design with Control Modules

David M. Robinson

University of Delaware, Newark, Delaware

Abstract

A number of researchers have considered asynchronous control networks as descriptive devices for theoretical studies of digital systems or as practical devices in the realization of digital systems. In this paper, the usual definition of an asynchronous control network is relaxed so that the control module itself need not be a fundamental mode logic. Under this assumption, it is shown that a universal control module may allow realization of most control networks. It is further shown that registers in the data flow structure of such systems may assume simple forms, often available MSI or LSI devices. A reasonable and cost effective design procedure is suggested by example.

Introduction

Recent researchers, such as Patil and Dennis [7] or Bruno and Altman [4], have considered the theoretical aspects of digital system description and realization using asynchronous control networks. These studies have established a formal description which allows a digital system designer to simplify design by segmenting a system into two interacting structures, one concerned with data flow and the other concerned with control. These recent efforts are indicative of a renewed interest in system design at the register-transfer level as presented by Bartee, et al. [1] in 1962. At least two practical implementations using register-transfer data flow structure concepts and distributed asynchronous control modules preceded the recent formalism; these are the "Macromodules" described by Clark [5] and the "Register-Transfer Modules" described by Bell [2].

In both the formal descriptions and the practical realizations, the network control modules consist of a family of fundamental mode circuits referred to as junctions, wyes, sequences, merges, K evokes, K subroutine calls, etc. In this paper, the requirement that the control module be a fundamental mode asynchronous circuit will be relaxed. The data flow structure may still be asynchronous; however, a clocked sequential machine will be realized for the implementation of control structures. This has significant implications. First, it is demonstrated that for a large class of digital systems, a single module may be utilized as a construct for all control networks. The universality of the module greatly simplifies the creation of cost effective control network designs. Second, it is shown that synchronization of the control module may result in considerable freedom in choosing the means used to realize the data flow structure of a digital system. Networks for the utilization of specific MSI or LSI devices may be configured.

A design procedure which yields documented, easily serviced and cost effective designs is suggested. A design example is given.

The Control Module

Reduced to its simplest form, the control module is a synchronous sequential machine with two

inputs, S and D, and two outputs, P and N (Fig. 1).

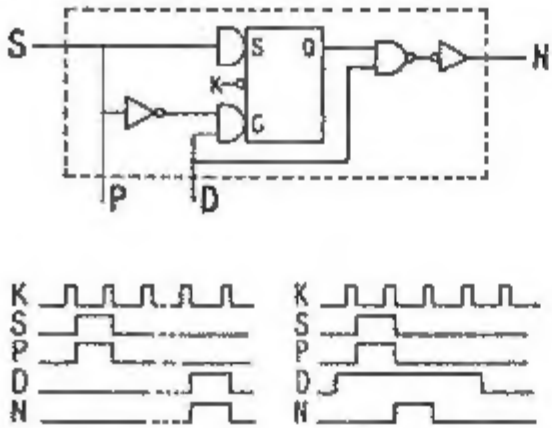


FIG. 1. SIMPLE CONTROL MODULE AND TIMING

A signal is applied at S to *start* the function of the module. The module supplies a signal P to the data flow structure of the system which initiates the *process* controlled by this particular module. This process may require an arbitrary number of clock intervals to run to completion, but when it has been completed, it supplies a signal D to the module indicating that the process function is *done*. The module then produces a signal N which may be used to start the *next* control module.

Several properties of the control module are:

1. The module can start itself (i.e., the output N can be returned to the input S) and the network will be stable in such a configuration.
2. The D signal can reside at logic high for all time and the module's operation will be the stable production of a clocked P-N signal sequence. Data flow registers which can guarantee completion in one clock interval need not compose a done signal.
3. There exists a duality of sorts between the P-N and the N-D signals. The characteristics of the signals are such that the P signal could start a module instead of the N signal or the N signal of a module could drive the D input of a module.
4. The P and N signals are levels which persist for one clock interval and may be used to set up gating in external registers which are then activated by clocks. Alternatively, the signals themselves may be used as clocks.

A more universally useful form of the module is shown in Fig. 2.

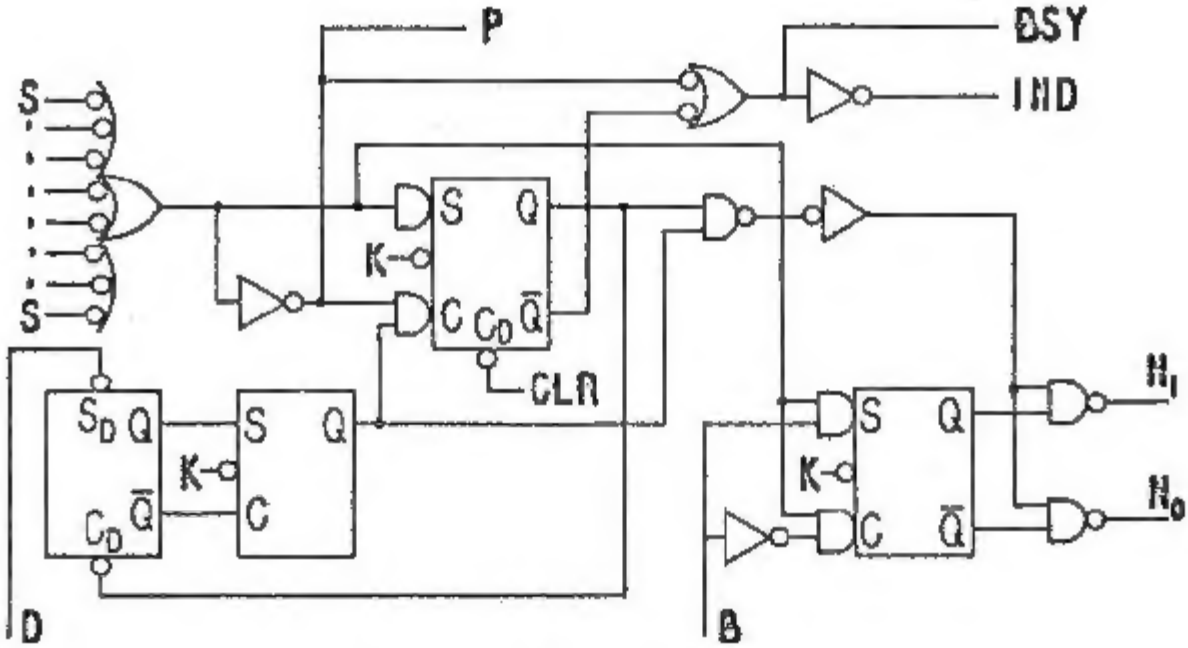


FIG. 2. UNIVERSAL CONTROL MODULE

Certain simple extensions have been added to the basic circuit:

1. A clear signal has been provided to permit initial control of the module (i.e., during turn-on).
2. The D signal input has been modified to allow an asynchronous data flow structure to be synchronized with the control structure.
3. A Boolean input signal (B, *Branch Direction*) has been introduced to allow steering the next output to either N_0 or N_1. The branch direction signal is sampled when this module is started. The signal may be produced asynchronously by the data flow structure.
4. Inputs and outputs have been modified under the assumption that the module will be connected in systems with low assertion logic levels.
5. Provision has been made to OR several signals to start the module.
6. As a diagnostic aid, a signal which indicates the status of the control module (BSY) is derived. This signal can drive an indicator in close proximity to the module (IND) or at some remote location.

For this particular realization, our fabrication techniques using 7400 series TTL allows four such modules on a 5"x8" printed circuit board. Clearly, the network is amenable to a technology which allows fabrication on a single chip.

Control Structures

Generally, the description of a digital system can be formulated using a flow chart. Other techniques are of course admissible, but designers seem to feel quite comfortable with a flow chart and control is easily followed using such a document. A desired control structure may then be visualized as a mapping from a flow chart into control module hardware. Several examples of control module interconnections which imitate flow chart constructions are given in Fig. 3.

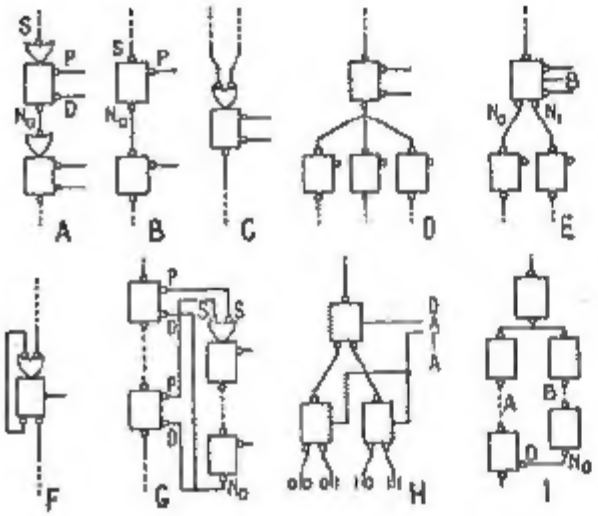


FIG. 3. CONTROL MODULE STRUCTURES

In Fig. 3-A, a simple sequence of events in controlled. As each process reaches completion, its done signal is returned to the module and the next module is started. For this situation, the branching facility of the module is not required and the branch direction lead is not shown in the diagram. In practice, that lead could be tied to ground and the next output taken from N_0. A similar notational simplification is implied in Fig. 3-B. Here, the done signal lead is not shown under the assumption that the processes are all single clock operations and the done signal may be simply set to 1 (ground). The complexity of each process is not of concern, only its timing. The process may be as simple as loading the contents of a register onto a data bus or as complex as performing a double precision multiply. If it can be accomplished in one clock interval, then the data flow structure need not construct the done signal. An additional economy of notation is introduced in 3-B where the OR input gate is omitted unless it is necessary in the description.

Fig. 3-C suggests the serial time merger of two processes may be accomplished by the OR operation. Multiple or parallel processes may be initiated by simply allowing the next function to start more than one control module as in 3-D. Procedures may be branched to selectively depending upon a value derived in the data flow structure as shown in Fig. 3-E.

Since the performance of the module is predictable even when it starts itself, the structure of Fig. 3-F is admissible and is useful for testing flags, allowing parallel merging, etc. Fig. 3-G demonstrates the ability of the module to call a sub-routine from a number of places int he main control sequence. Here, one should note that the usual role of the P output has been modified so that it starts a module and the N signal is employed as an input to a D.

A multiple branch may be implemented as the tree structure of Fig. 3-H. Generally, this would not be a very cost-effective solution to a multiple branching problem but it does support the universal nature of the module. Two existing chips (a quad-latch and a one of sixteen decoder) can easily extend the module capabilities for multiple branching.

Fig. 3-I suggests a simple merging pattern which is allowed under certain restrictions. If one has apriori knowledge that the time to complete procedure A is less than the time to complete procedure B, the technique is appropriate. If A were to require more time than B, the network would not be speed independent and would likely malfunction. If the timing situation is unclear, flags int he data flow structure may be set when the two procedures are initiated and the structure

of Fig. 3-F can be used to accomplish the requisite merge when timing is reconciled.

In short, many useful control structures may be realized using the module. Of course, the module does not solve any problems associated with resource contention or other malformed flow charts; that is the function of the designer. It may be possible to hypothesize cases which are impossible to solve using only this module, and it is certainly possible to find cases in which the module alone may not be the more effective solution (i.e., the branching tree). However, for many real digital systems, this single control module allows a designer substantial freedom in implementing the system control structure.

Registers for the Data Flow Structure

A set of generalized register-transfer modules is not a prerequisite for design with this control module. In fact, such a set might be self-defeating. Experience suggests that register design for a data flow structure is quite straightforward for systems which employ this type of synchronous control module. Many registers may be existing MSI and LSI devices. Indeed, a strong motivation for the philosophy of the distributed control scheme is to provide a simple design procedure which allows a designer to implement a digital system using larger logic modules at the register level. Most available MSI and LSI devices have been produced in response to the requirements of the data flow structure rather than to the requirements of control (a notable exception is the case of ROMs).

An examination of system design procedures yields further justification for avoiding a specific set of register-transfer modules. It is difficult to identify a general design procedure. However, it is often the case that a modest design problem might be approached by assuming or describing the data flow structure of the system. This may be a formal procedure which identifies the required registers as in a register transfer language or in a data flow graph. The designer then identifies the control signals which are available to direct the controller and the signals which the controller must provide to the data flow structure. The designer may then proceed with control realization using state diagrams, assignment algorithms, simplification techniques, etc. As the system grows in complexity, he is forced to abandon this procedure and may decide to microprogram his control. In many cases, he gives up completely and installs a programmable controller in the system. These may not be very cost-effective solutions.

In contrast to the above procedure, which the concept of a control module, the designer may begin by developing a program or a flow chart. The exercise of setting down the flow chart usually partitions the system into manageable data flow structure elements. The usually difficult design, the control, is a one-to-one mapping of the flow chart into hardware. Hence, there is no real need for a family of data flow structure modules. He makes optimum use of the available state of the integrated circuit art and he may simply configure his controller to take advantage of new developments as they arise.

A Simple Design Example

A bench mark problem for control module design seems to have been established by precedent. The problem of summing positive integers from 1 to N has been solved using RTMs by Bell [3]. Ellis and Franklin [6] compared Bell's solution with one derived using the Macromodules of Clark. The algorithm to be implemented is stated in flow chart form in Fig. 4.

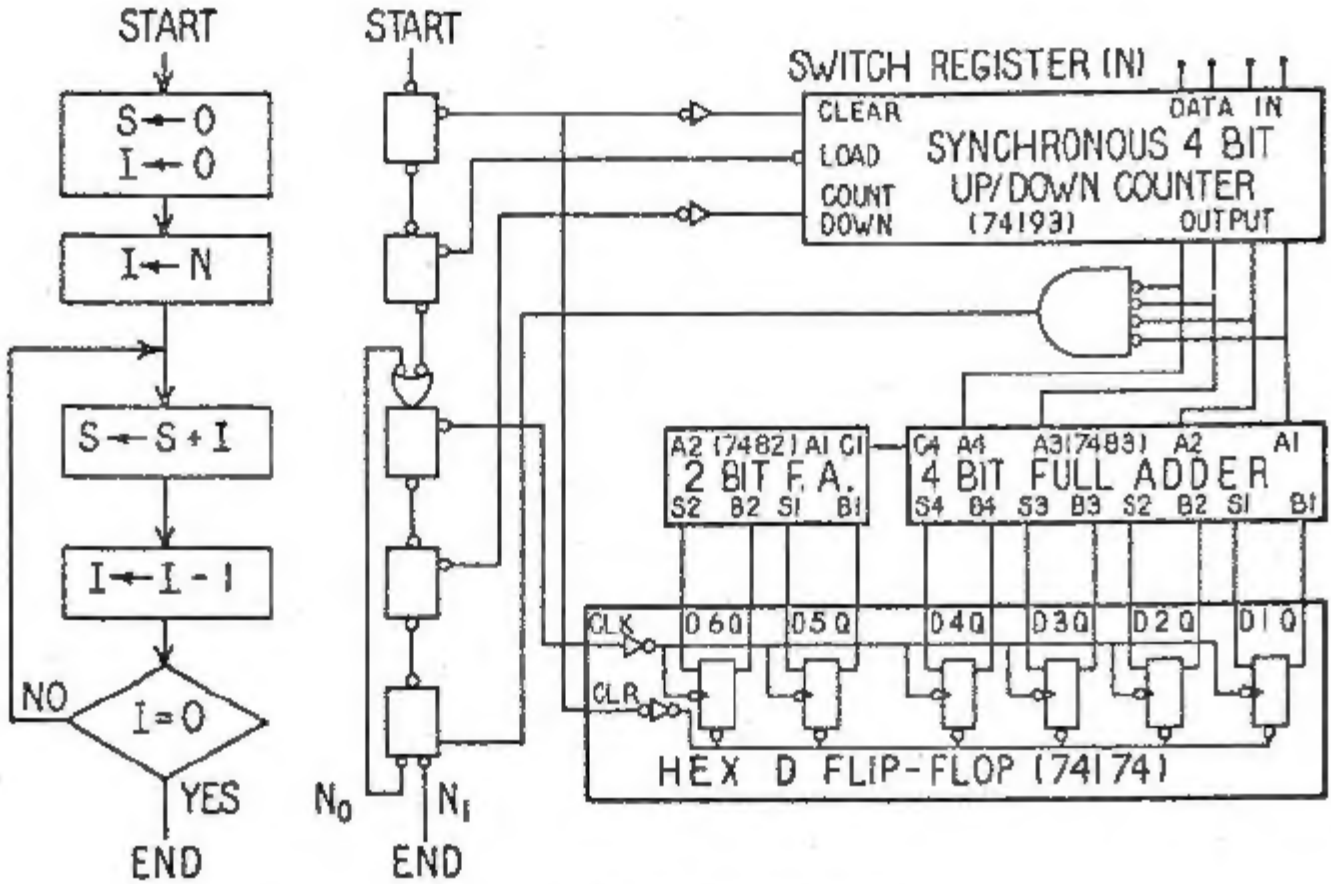


FIG. 4. SUM OF INTEGERS DESIGN EXAMPLE

Clearly, setting down the flow chart has identified a number of characteristics of the data flow structure of this system. There is a register I which one must be able to clear and to set to a number N that is available from a set of switches. This register must also be capable of decrementing its contents and the control structure must be able to determine when it has reached zero. That description suggests a pre-settable binary down counter (a single MSI function) with its clear state decoded. Additionally, the data flow structure requires a register S which may be cleared and loaded on command with the sum of the S and I registers. Again, MSI functions are available to relived the designer of much of his burden in the data structure portion of this system. The total implementation of the system is shown in Fig. 4. The isomorphism between the control hardware and the flow chart is obvious, and the interaction between the control and data flow structures needs no explanation.

The implementation of Fig. 4 requires three clock times for the looped operation, hence, its timing is clear and requires 3 microseconds for a 1 MHz clock or 300 nanoseconds for a 10 MHz clock, etc. Of course, maximum clocking rates must be determined by operations times in the data flow structure (i.e., carry propagation time for the binary full adder, etc.) This implementation can be operated at 10 MHz. The reference cited (6) gives execution times for the loop in this algorithm of 1.8 microseconds using RTMs and 1.1 microseconds using MMs.

Some Fringe Benefits

The design time for systems of this sort is reasonably minimal. It is agreed that a special purpose controller with minimized internal state configuration could require fewer chips in its

implementation than this distributed control, but that cost advantage might easily be offset by design effort. Further, one should not discount the possibility that the control module could be integrated to a single chip.

The system function is clear; documentation of the system is well started; de-bugging techniques and servicing procedures are clearly suggested, all as a result of the direction which has been taken in the design procedure.

Acknowledgments

The author would like to acknowledge G. Rossmann and D. Grim for their contributions to this effort.

References

1. Bartee, T.C., Lebow, I.L., and Reed, I.S., Theory and Design of Digital Machines, McGraw-Hill, 1962.
2. Bell, C.G. and Grason, J., "The Register Transfer Module Concept," Computer Design, May 1971.
3. Bell, C.G., et al., "The Description and Use of Register Transfer Modules, " IEEE Trans., C-21#5, 1972.
4. Bruno, J. and Altman, S.M., "Theory of Asynchronous Control Networks, " IEEE Trans., C-20#6, 1971.
5. Clark, W.A., "Macromodular Computer Systems." Spring Joint Computer Conference 1967.
6. Ellis, R.A. and Franklin, J.A., "High-Level Logic Modules: A Qualitative Comparison," Proceedings Compcon 72 Conferences, Sept. 1972.
7. Patil, S.S. and Dennis, J.B., "The Description and Realization of Digital Systems," Proceedings Compcon 72 Conference, Sept. 1972.